Understanding Network Performance

SC2001 Tutorial S8

11 November 2001

Phillip Dykstra
Chief Scientist
WareOnEarth Communications, Inc.
phil@sd.wareonearth.com

Motivation

If our networks are so fast, how come my ftp is so slow?

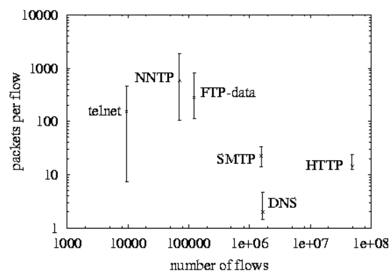
Objectives

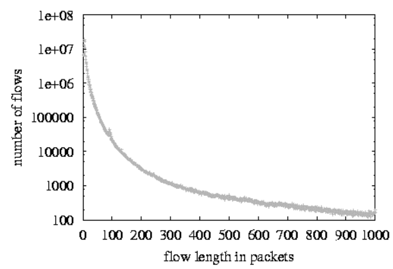
- Learn what is required for high speed data transfer and what to expect
- Fundamental understanding of delay, loss, bandwidth, routes, MTU, windows
- Examine TCP dynamics
- Look at basic tools and what they tell you
- Provide background for S12, “Achieving Network Performance”

Unique HPC Environment

- The Internet is being optimized for:
  - millions of users behind low-speed soda straws
  - thousands of high-bandwidth servers serving millions of soda straw streams
- Single high-speed to high-speed flows get little commercial attention

What's on the Internet?

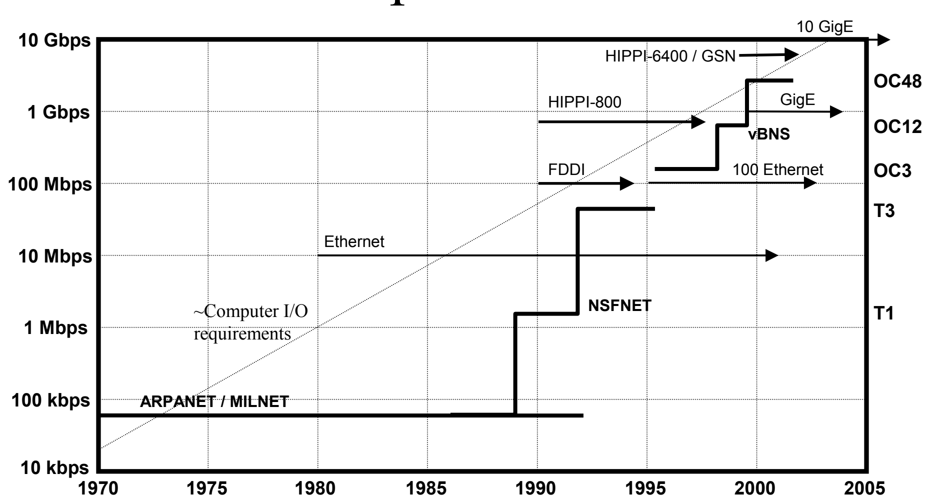




- Well over 90% of it is TCP; most of that is Web
- Most flows are less than 30 packets long

InternetMCI, 1998, k. claffy

Network Speeds Over Time

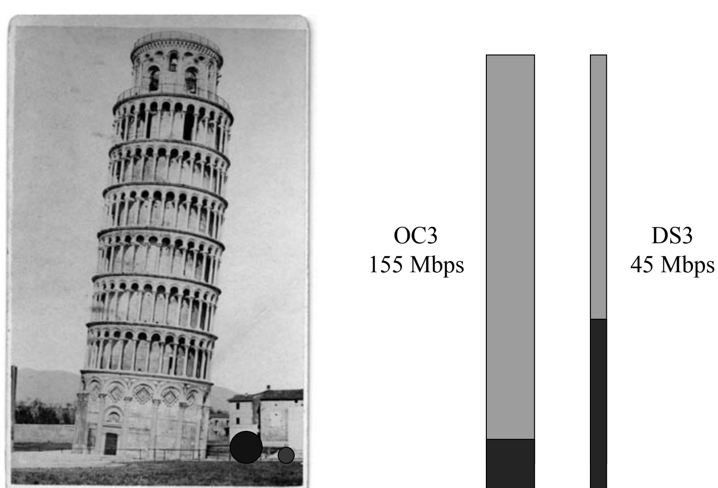


Delay

a.k.a. Latency

Capacity

High "~~Speed~~" Networks



Speed of Light in Media

- ~3.0×10^8 m/s in free space
- ~2.3×10^8 m/s in copper
- ~2.0×10^8 m/s in fiber = 200 km / ms

  [100 km of distance = 1 ms of round trip time]

Packet Durations and Lengths

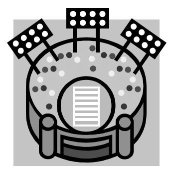
1500 Byte Packets in Fiber

	Mbps	pps	sec/pkt	length
56k	0.056	4.7	214 ms	42857 km
T1	1.544	129	7.8 ms	1554 km
Eth	10	833	1.2 ms	240 km
T3	45	3750	267 us	53 km
FEth	100	8333	120 us	24 km
OC3	155	13k	77 us	15 km
OC12	622	52k	19 us	3859 m
GigE	1000	83k	12 us	2400 m
OC48	2488	207k	4.8 us	965 m
10GigE	10000	833k	1.2 us	240 m

Observations on Packet Lengths

- A 56k packet could wrap around the earth!

- A 10GigE packet fits in the convention center



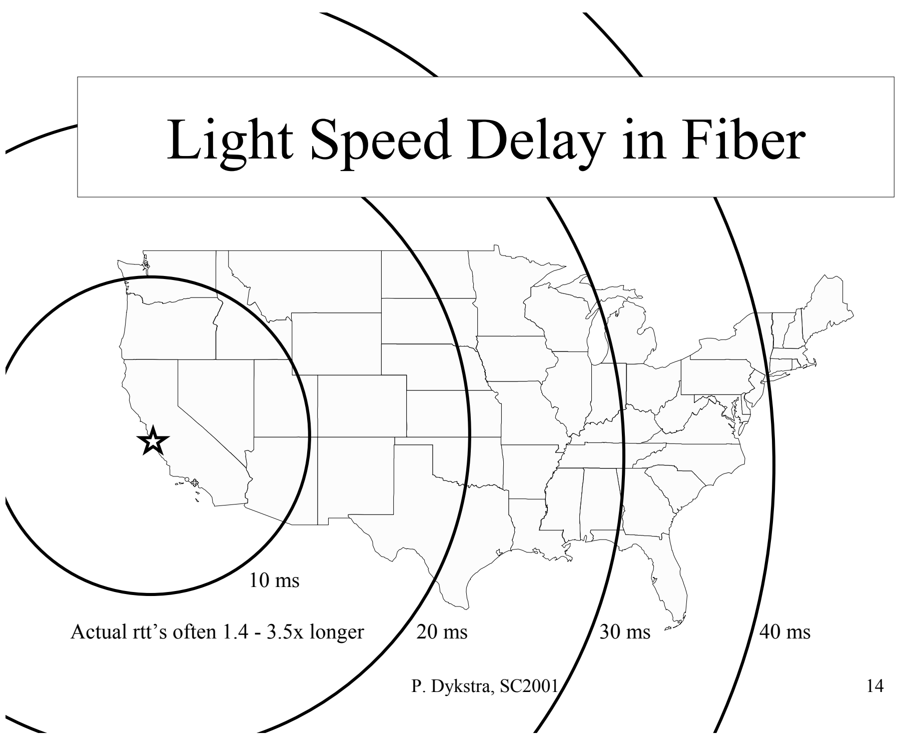
Observations on Packet Lengths

- Each store and forward router hop adds the packet duration to the delay
  - In the old days (< 10 Mbps) such hops dominated delay
  - Today (> 10 Mbps) store and forward delays on WANs are minimal compared to propagation

Observations on Packet Lengths

- ATM cells (and TCP ACK packets) are ~1/30th as long, 30x as many per second
  - One of the reasons we haven't seen OC48 SAR
- Jumbo Frames (9000 bytes) are 6x longer, 1/6th as many per second

Light Speed Delay in Fiber



Measuring Delay - Ping

```
% ping -s 56 sgi.com
PING sgi.com (192.48.153.65) from 63.196.71.246 : 56(84) bytes of data.
64 bytes from SGI.COM (192.48.153.65): icmp_seq=1 ttl=240 time=31.6 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=2 ttl=240 time=66.9 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=3 ttl=240 time=33.4 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=4 ttl=240 time=36.7 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=5 ttl=240 time=40.9 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=6 ttl=240 time=104.8 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=7 ttl=240 time=177.5 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=8 ttl=240 time=34.2 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=9 ttl=240 time=31.5 ms
64 bytes from SGI.COM (192.48.153.65): icmp_seq=10 ttl=240 time=31.9 ms

--- sgi.com ping statistics ---
11 packets transmitted, 10 packets received, 9% packet loss
round-trip min/avg/max = 31.5/58.9/177.5 ms
```


Ping Observations

- Ping packet = 20 bytes IP + 8 bytes ICMP + "user data" (first 8 bytes = timestamp)
- Default = 56 user bytes = 64 byte IP payload = 84 total bytes
- Small pings (-s 8 = 36 bytes) take less time than large pings (-s 1472 = 1500 bytes)

Ping Observations

- TTL = 240 indicates 255-240 = 15 hops
- Delay variation indicates congestion or system load
- Not good at measuring small loss
  - An HPC network should show zero ping loss
- Depends on ICMP ECHO which is sometimes blocked for “security”

Bandwidth*Delay Product

- The number of bytes in flight to fill the entire path
- Includes data in queues if they contributed to the delay
- Example
  - 100 Mbps path
  - ping shows a 75 ms rtt
  - BDP = 100 * 0.075 = 7.5 million bits (916 KB)

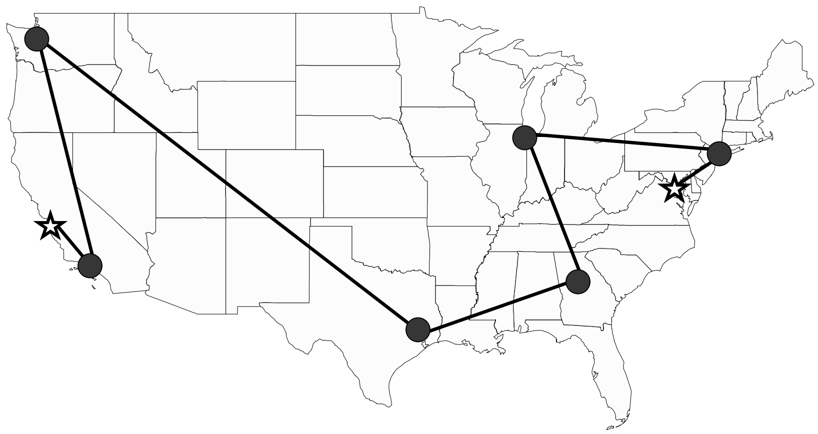
Routes

The path taken by your packets

How Routers Choose Routes

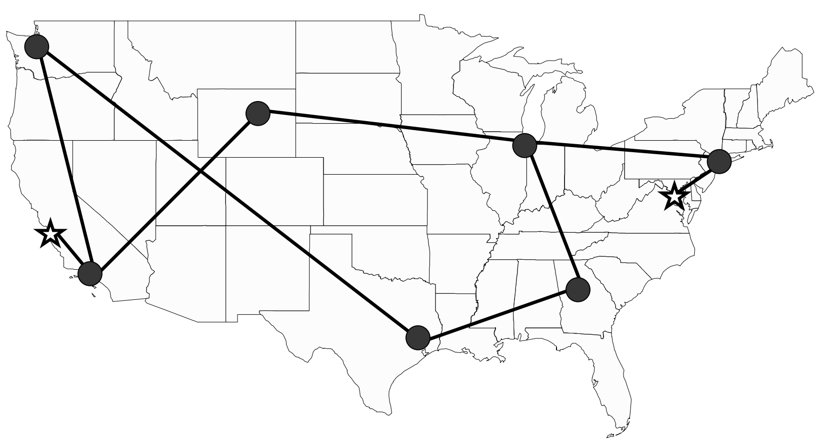
- Within a network
  - Smallest number of hops
  - Highest bandwidth paths
  - Usually ignore latency and utilization
- From one network to another
  - Often “hot potato” routing, i.e. pass to the other network ASAP

“Scenic” Routes

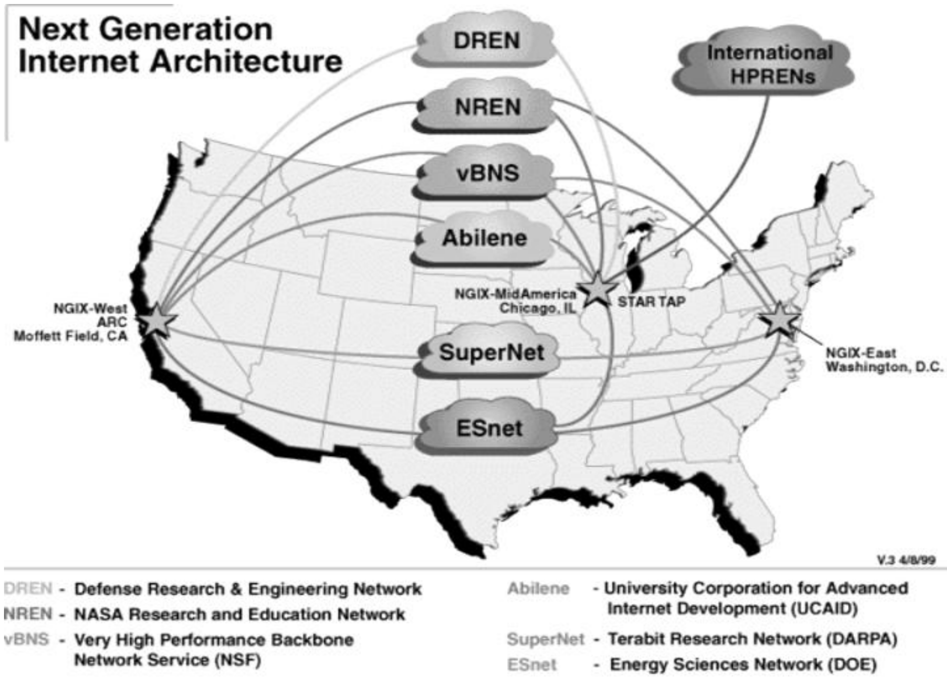


P. Dykstra, SC2001 21

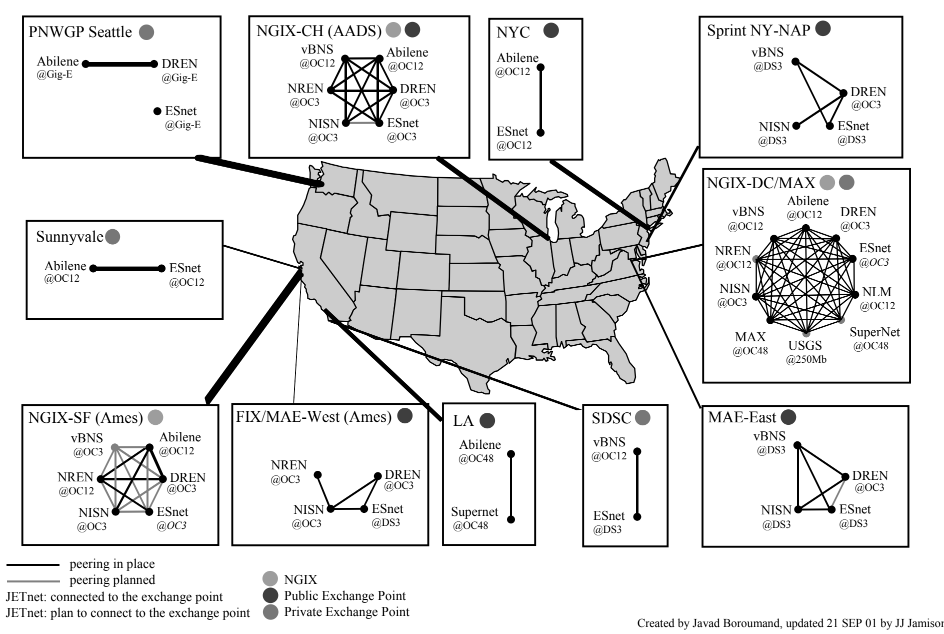
Asymmetric Routes



P. Dykstra, SC2001 22



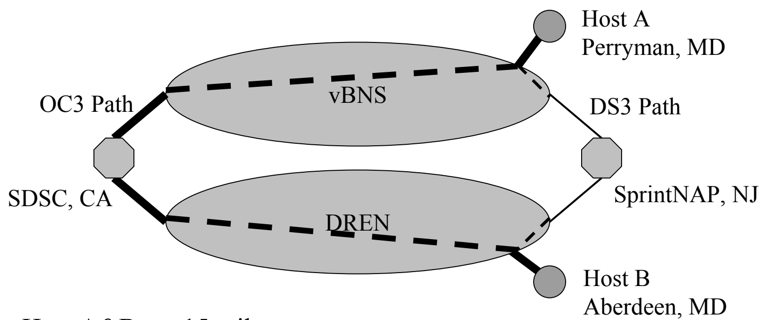
JETnets Interconnections and Peering



Created by Javad Boroumand, updated 21 SEP 01 by JJ Jamison

Path Performance: Latency vs. Bandwidth

The highest bandwidth path is not always the highest throughput path!

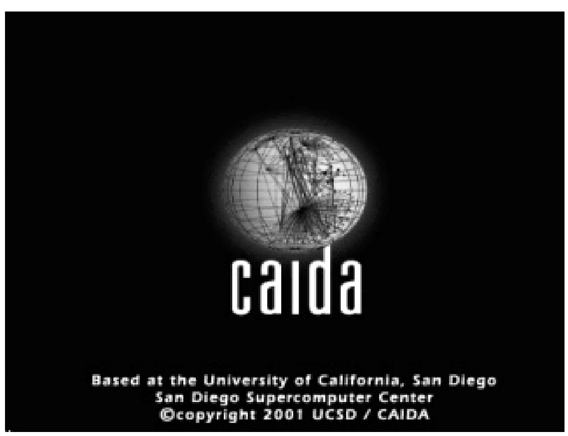


- Host A&B are 15 miles apart
- DS3 path is ~250 miles
- OC3 path is ~6000 miles

The network chose the OC3 path with 24x the rtt, 80x BDP

How Traceroute Works

www.caida.org/outreach/resources/animations/



Traceroute Observations

- Shows the return interface addresses of the **forwarding** path
- You can't see hops through switches or over tunnels (e.g. ATM VC's, GRE, MPLS)
- Depends on ICMP TTL Exceeded
  - Sometimes blocked for "security"
- Final hop depends on ICMP Port Unreachable
  - Sometimes blocked for "security"

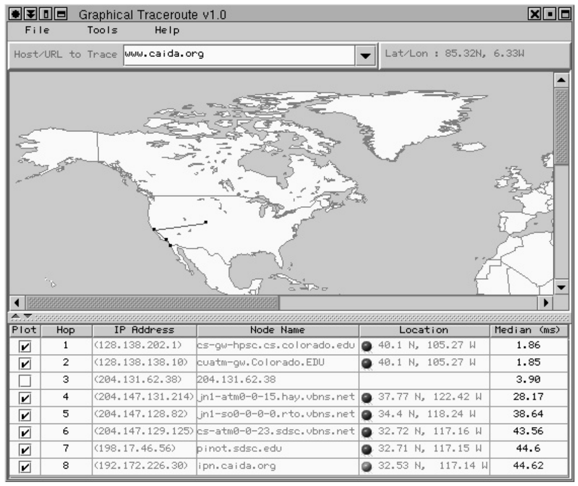
Matt's Traceroute

www.bitwizard.nl/mtr/

```
                          Matt's traceroute  [v0.41]
damp-ssc.spawar.navy.mil                                Sun Apr 23 23:29:51 2000
Keys:  D - Display mode    R - Restart statistics    Q - Quit
                                           Packets               Pings
Hostname                                %Loss  Rcv   Snt   Last Best  Avg  Worst
 1. taco2-fe0.nci.net                      0%   24    24      0    0    0      1
 2. nccosc-bgp.att-disc.net                0%   24    24      1    1    1      6
 3. pennsbr-aip.att-disc.net               0%   24    24     84   84   84     86
 4. sprint-nap.vbns.net                    0%   24    24     84   84   84     86
 5. cs-hssi1-0.pym.vbns.net                0%   23    24     89   88  152    407
 6. jn1-at1-0-0-0.pym.vbns.net             0%   23    23     88   88   88     90
 7. jn1-at1-0-0-13.nor.vbns.net            0%   23    23     88   88   88     90
 8. jn1-so5-0-0-0.dng.vbns.net             0%   23    23     89   88   91    116
 9. jn1-so5-0-0-0.dnj.vbns.net             0%   23    23    112  111  112    113
10. jn1-so4-0-0-0.hay.vbns.net             0%   23    23    135  134  135    135
11. jn1-so0-0-0-0.rto.vbns.net             0%   23    23    147  147  147    147
12. 192.12.207.22                          5%   22    23     98   98  113    291
13. pinot.sdsc.edu                         0%   23    23    152  152  152    156
14. ipn.caida.org                          0%   23    23    152  152  152    160
```

GTrace – Graphical Traceroute

www.caida.org/tools/visualization/gtrace/



Path MTU

- Maximum Transmission Unit (MTU)
  – Largest packet that can be sent as a unit
- Path MTU
  – min MTU of all hops in a path
- Hosts can do Path MTU Discovery to find it
  – Depends on ICMP replies
- Without PMTU Discovery should assume it's only 576 bytes
  – Some hosts falsely assume 1500

Bandwidth

and throughput

Throughput Limit

- throughput <= **available** bandwidth

  (link with the minimum unused bandwidth)

  – A high performance network should be lightly loaded (<50%?)

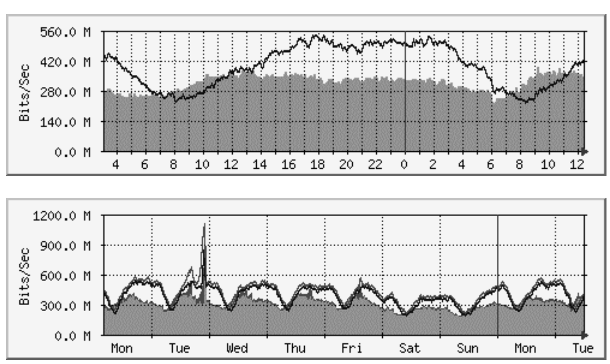
  – *A loaded high speed network is no better to the end user than a lightly loaded slow one*

MRTG
MULTI ROUTER TRAFFIC GRAPHER

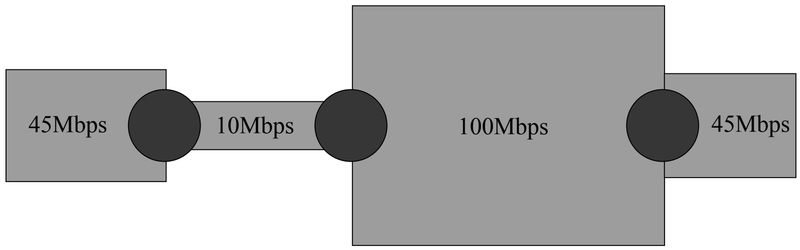
- www.mrtg.org
- Extremely popular network monitoring tool
- Most common display:
  - Five minute average link utilizations
  - Green into interface
  - Blue out of interface
- RRDTool newer generalized version (same site)

MRTG Example

Abilene, Kansas City to Denver OC48 link, 9 October 2001

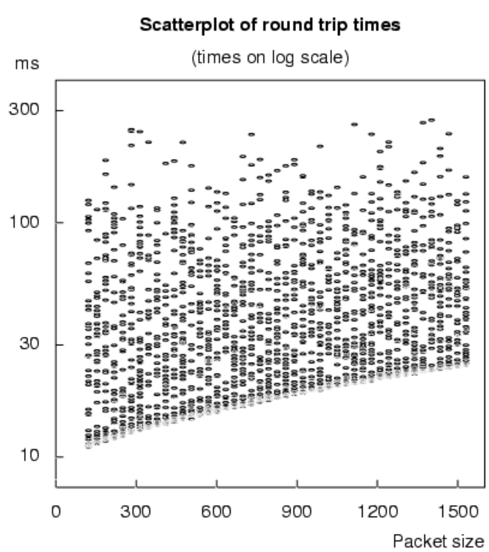


Hops of Different Bandwidth



- The "Narrow Link" has the lowest bandwidth
- The "Tight Link" has the least **Available** bandwidth
- Queues can form wherever available bandwidth decreases
- A queue buildup is most likely in front of the Tight Link

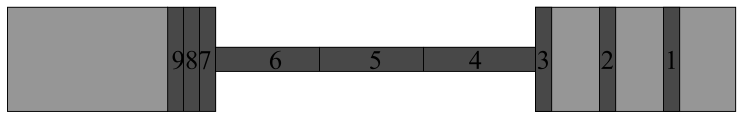
Bandwidth Estimation – Single Packet



- Larger packets take longer
- Delay from intercept
- Bandwidth from slope

From A. Downey

Bandwidth Estimation – Multi Packet



- Packet pairs or trains are sent
- The slower link causes packets to spread
- The packet spread indicates the bandwidth

Bandwidth Measurement Tools

- pathchar – Van Jacobson, LBL
  - ftp://ftp.ee.lbl.gov/pathchar/
- clink – Allen Downey, Wellesley College
  - http://rocky.wellesley.edu/downey/clink/
- pchar – Bruce A. Mah, Sandia/Cisco
  - http://www.employees.org/~bmah/Software/pchar/

Bandwidth Measurement Tools

- pipechar - Jin Guojun, LBL
  - http://www.didc.lbl.gov/pipechar/
- nettimer - Kevin Lai, Stanford University
  - http://gunpowder.stanford.edu/~laik/projects/nettimer/
- pathrate - Constantinos Dovolis, Univ of Delaware
  - http://www.cis.udel.edu/~dovrolis/bwmeter.html

Treno Throughput Test

www.psc.edu/networking/treno_info.html

- Tells you what a good TCP should be able to achieve (Bulk Transfer Capacity)

```
damp-mhpcc%  treno damp-pmrf
 MTU=8166  MTU=4352  MTU=2002  MTU=1492 ..........
Replies were from damp-pmrf [192.168.1.1]
   Average rate: 63470.5 kbp/s (55241 pkts in + 87 lost = 0.16%) in 10.03 s
Equilibrium rate: 63851.9 kbp/s (54475 pkts in + 86 lost = 0.16%) in 9.828 s
Path properties: min RTT was   8.77 ms, path MTU was 1440 bytes
```

Treno Observations

- Easy 10 second test, no remote access or receiver process required
- Emulates TCP but doesn't use TCP
  - Problems with host TCP or tuning are avoided
- Does Path MTU Discovery
- Reports rtt and loss rates
- A zero equilibrium result means there was too much packet loss to exit "slow start"

Treno Observations

- Can send ICMP (`-i`) or UDP (default)
  - ICMP replies (ECHO or UNREACH) could be blocked for "security"
- Routers send ICMP replies very slowly
  - So don't test routers with treno
- ICMP is often rate limited now by hosts

TCP Throughput Tests

- ttcp – the original, many variations
  - http://sd.wareonearth.com/~phil/net/ttcp/
- Iperf – great TCP/UDP tool (recommended)
  - http://dast.nlanr.net/Projects/Iperf/
- netperf – dated but still in wide use
  - http://www.netperf.org/
- ftp – nothing beats a real application

Throughput Testing Notes

- Network data rates (bps) are powers of 10, not powers of 2 as used for Bytes
  - E.g. 100 Mbps ethernet is 100,000,000 bits/sec
  - Some tools wrongly use powers of 2 (e.g. ttcp)
- User payload data rates are reported by tools
  - No TCP, IP, Ethernet, etc. headers are included
  - E.g. 100 Mbps ethernet max is 97.5293 Mbps
    - http://sd.wareonearth.com/~phil/net/overhead/

Windows

Flow/rate control and error recovery

Window Sizes 1,2,3

Data packets go one way
ACK packets come back

TCP Throughput

- Rate = **window** / rtt

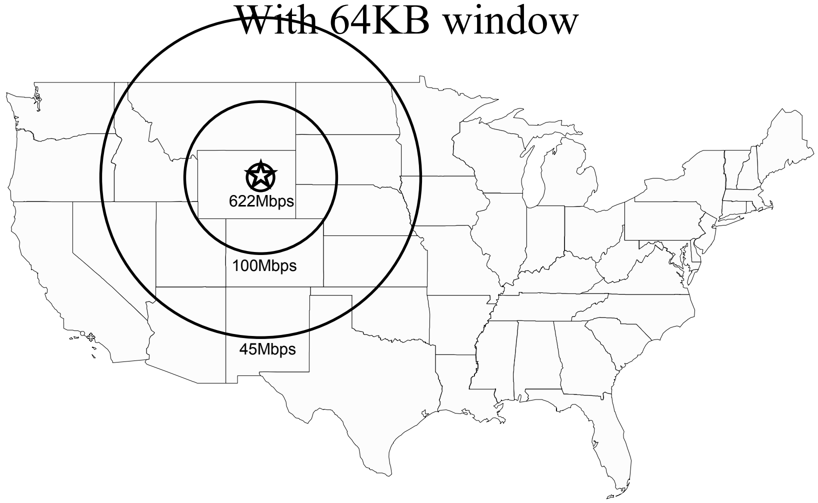
  window = min(send_buf, rwin, cwin)

  cwin =~ 0.7 * MSS / sqrt(pkt_loss)

- Receive window (rwin) and/or send_buf are still the most common performance limiters
  - E.g. 8kB window, 87 msec ping time = 753 kbps
  - E.g. 64kB window, 14 msec rtt = 37 Mbps

Maximum TCP/IP Data Rate

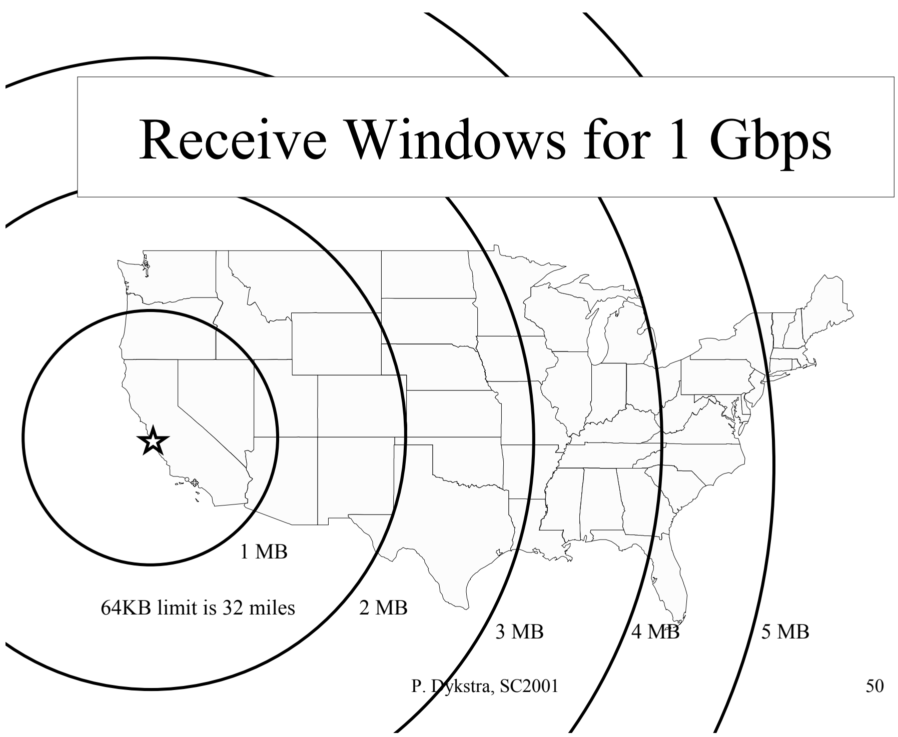
With 64KB window

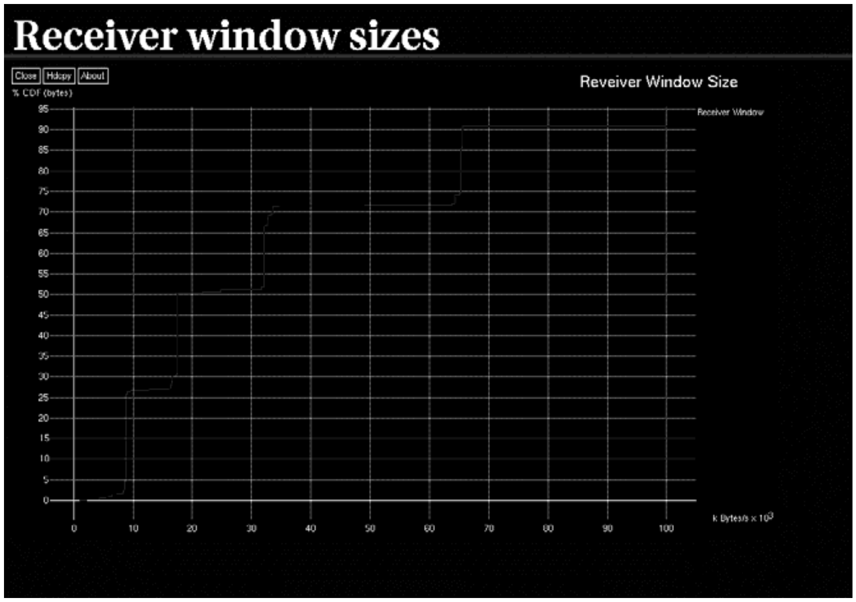


Bandwidth*Delay Product and TCP

- TCP needs a **receive window** (rwin) equal to or greater than the BW*Delay product to achieve maximum throughput
- TCP needs **sender side socket buffers** of 2*BW*Delay to recover from errors
- You need to send about 3*BW*Delay bytes for TCP to reach maximum speed

Receive Windows for 1 Gbps





P. Dykstra, SC2001 51

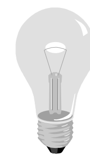
Observed Receiver Window Sizes

- ATM traffic from the Pittsburgh Gigapop
- 50% have windows < 20 KB
  - These are obsolete systems!
- 20% have 64 KB windows
  - Limited to ~ 8 Mbps coast-to-coast
- ~9% are assumed to be using window scale

M. Mathis, PSC

P. Dykstra, SC2001 52

Things You Can Do

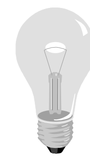


- Find out the rtt with ping, compute BDP
- Make sure your HPC apps offer sufficient receive windows and use sufficient send buffers
  - But don’t run your system out of memory

System Tuning

buffers, windows, etc.

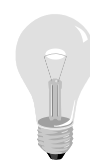
Things You Can Do



- Throw out your low speed interfaces and networks! 
- Make sure routes and DNS report high speed interfaces
- Don’t over-utilize your links (<50%?)
- Use routers sparingly, host routers not at all

```
routed -q
```


Things You Can Do



- “Do the math” i.e. know what kind of throughput and loss to expect for your situation
- Check your TCP for high performance features
- “Tune” your system
  - http://www.psc.edu/networking/perf_tune.html
- Look for sources of loss
  - Watch out for duplex problems (late collisions?)

FreeBSD Tuning

```
# FreeBSD 3.4 defaults are 524288 max, 16384 default
/sbin/sysctl -w kern.ipc.maxsockbuf=1048576
/sbin/sysctl -w net.inet.tcp.sendspace=32768
/sbin/sysctl -w net.inet.tcp.recvspace=32768
```

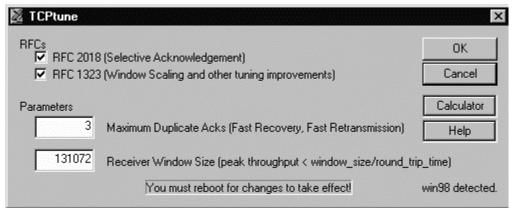

Linux 2.4 Tuning

`/etc/sysctl.conf`

```
# Increase max socketbuffer sizes, actual = 2x these values
net.core.rmem_max = 1048576
net.core.wmem_max = 1048576

net.ipv4.icmp_echoreply_rate = 0
net.ipv4.icmp_destunreach_rate = 0
net.ipv4.ip_no_pmtu_disc = 0
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 1
```

TCPTune

A TCP Stack Tuner for Windows



- http://moat.nlanr.net/Software/TCPtune/
- Makes sure high performance parameters are set
- Many such utilities for **modems**, e.g. DunTweak, but they reduce performance on high speed networks

Ethernet Duplex Problems

An Internet Epidemic!

- Ethernet “auto-negotiation” can select the speed and duplex of a connected pair
- If only one end is doing it:
  - It can get the speed right
  - It will assume **half-duplex**
- Mismatch only shows up under load
  - Can’t see it with ping

TCP

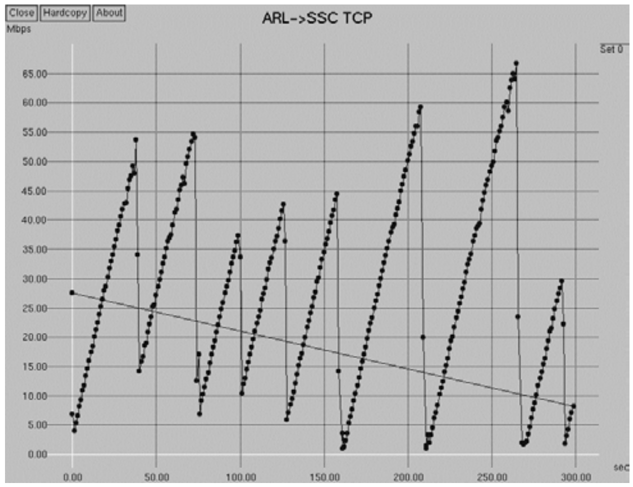
The Internet’s transport

Important Points About TCP

- TCP is *adaptive*
- It is *constantly* trying to go *faster*
- It always *slows down* when it detects a *loss*

- *How much* it sends is controlled by *windows*
- *When* it sends is controlled by *received ACK’s* (or timeouts)

TCP Throughput vs. Time



TCP Throughput

Once recv window size and available bandwidth aren't the limit

$$\text{Rate} \sim= \frac{0.7 * \text{Max Segment Size (MSS)}}{\text{Round Trip Time (latency)} \quad \text{sqrt[pkt\_loss]}}$$

M. Mathis, et al.

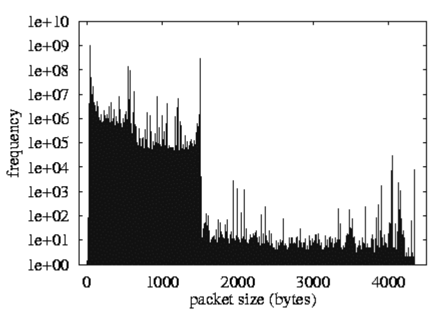
- Double the MTU, double the throughput
- Halve the latency, double the throughput
  - shortest path matters
- Halve the loss rate, 40% higher throughput

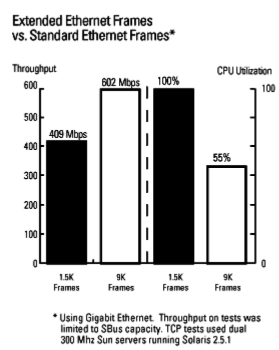
Max Segment Size (MSS)

rate = 0.7 * **MSS** / (rtt * sqrt(p))

- MSS = MTU – packet headers
- Common MTU's
  - 576 IPv4 default
  - 1500 ethernet, IPv6 default
  - ~9000 GigE Jumbo Frame, CLIP ATM
  - 64k max ATM AAL5 frame
- Jumbo frame => ~6x throughput increase

Packet Size (MTU) Issues

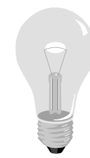




http://sd.wareonearth.com/~phil/jumbo.html

"New York to Los Angeles. Round Trip Time (rtt) is about 40 msec, and let's say packet loss is 0.1% (0.001). With an MSS of 1460 bytes, TCP throughput will have an upper bound of about 6.5 Mbps! And no, that is not a window size limitation, but rather one based on TCP's ability to detect and recover from congestion (loss). With 9000 byte frames, TCP throughput could reach about 40 Mbps."

Things You Can Do



- Use only large MTU interfaces/routers/links
  - Gigabit Ethernet with **Jumbo Frames** (9000)
  - ATM CLIP (9180)
- Never reduce the MTU (or bandwidth) on the path between each/every host and the WAN
- Make sure your TCP uses Path MTU Discovery

Round Trip Time (RTT)

rate = 0.7 * MSS / (**rtt** * sqrt(p))

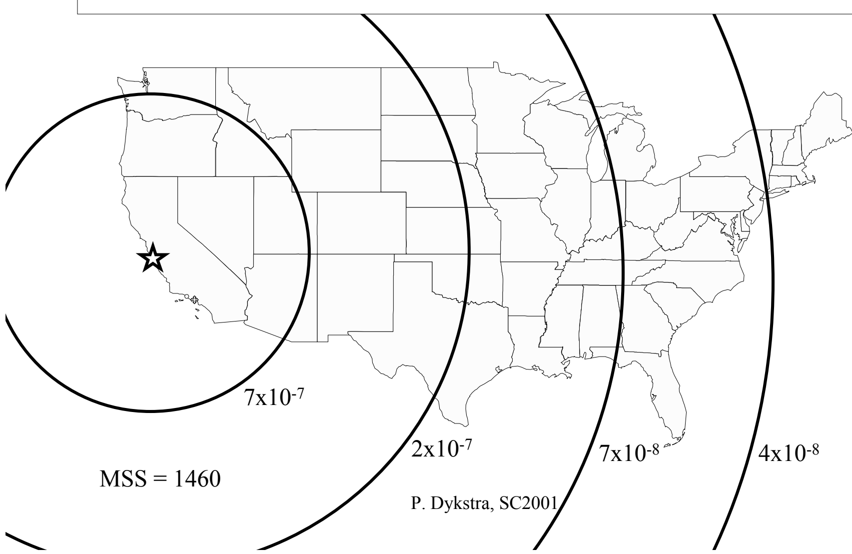
- If we could halve the delay we could double throughput!
- Most delay is caused by speed of light in fiber (~200 km/msec)
- “Scenic routing” and fiber paths raise the minimum
- Congestion (queuing) adds delay

Packet Loss (p)

rate = 0.7 * MSS / (rtt * sqrt(**p**))

- ***Loss dominates throughput***
- At least 6 orders of magnitude observed on the Internet
- 100 Mbps throughput requires $O(10^{-6})$
- 1 Gbps throughput requires $O(10^{-8})$

Loss Limits for 1 Gbps



More About TCP

Some details

TCP Keeps Evolving

- TCP, RFC793, Sep 1981
- Reno, BSD, 1990
- Path MTU Discovery, RFC1191, Nov 1990
- Window Scale, PAWS, RFC1323, May 1992
- SACK, RFC2018, Oct 1996
- NewReno, April 1999
- More on the way!

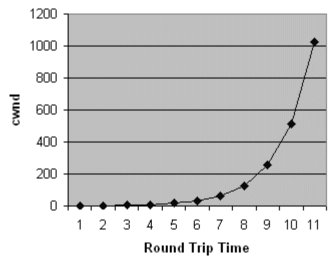
TCP Reno

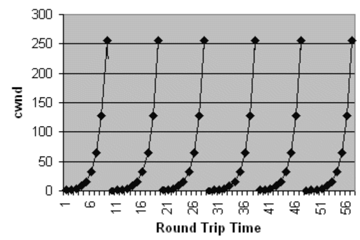
- Most modern TCP’s are “Reno” based
- Reno defined (refined) four key mechanisms
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
  - Fast Recovery
- NewReno refined fast retransmit/recovery when partial acknowledgements are available

TCP Congestion Window

- Congestion window (cwnd) controls startup and limits throughput in the face of loss.
- cwnd gets larger after every new ACK
- cwnd get smaller when loss is detected
- Usable window = min(rwin, cwnd)

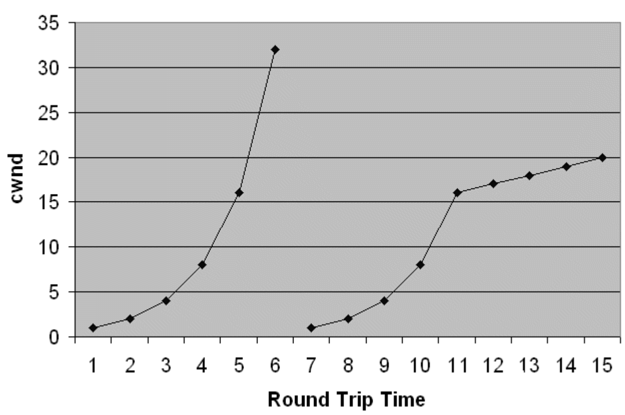
Cwnd During Slowstart





- cwnd increased by one for every new ACK
- cwnd doubles every round trip time
- cwnd is reset to zero after a loss

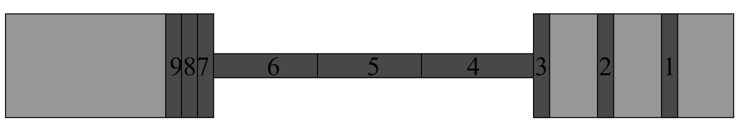
Slowstart and Congestion Avoidance Together



Delayed ACKs

- TCP receivers send ACK's:
  – after every second segment
  – after a delayed ACK timeout
  – on every segment after a loss (missing segment)
- A new segment sets the delayed ACK timer
  – Typically 0-200 msec
- A second segment (or timeout) triggers an ACK and clears the delayed ACK timer

ACK Clocking



- A queue forms in front of a slower speed link
- The slower link causes packets to spread
- The spread packets result in spread ACK's
- The spread ACK's end up clocking the source packets at the slower link rate

Detecting Loss

- Packets get discarded when queues are full (or nearly full)
- Duplicate ACK's get sent after missing or out of order packets
- Most TCP's retransmit after the third duplicate ACK ("triple duplicate ACK")

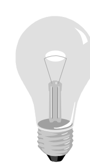
Random Early Detection (RED)

- Discards arriving packets as a function of queue length
- Gives TCP better congestion indications (drops)
- Avoids "Global Synchronization"
- Increases total number of drops
- Increases link utilization
- Many variations (weighted, classed, etc.)

SACK TCP
Selective Acknowledgement

- Specifies exactly which bytes were missed
- Better measures the “right edge” of the congestion window
- Can do a **very** good job keeping your queues full
  - Which causes latencies to go way up
- Without RED, will cause global sync faster
- Win98, Win2k, Linux have SACK

Things You Can Do



- Consider using RED on your routers before wide scale deployment of SACK TCP
- SACK won’t care very much but your old TCP’s will thank you
- Consider a priority class of service for interactive traffic?

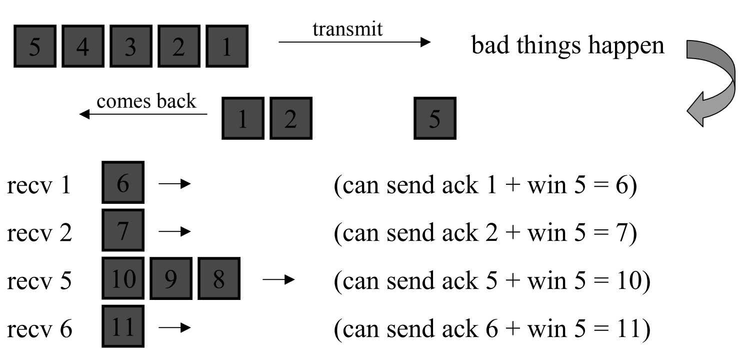
Advanced Debugging

Mping

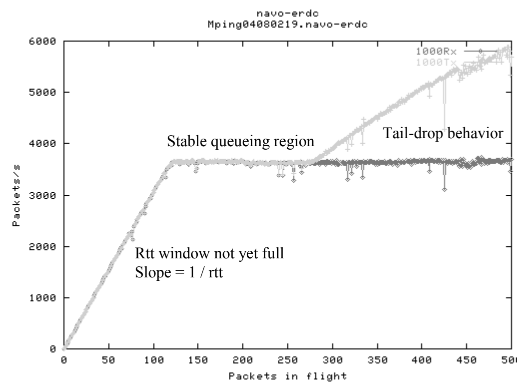
MPing - A Windowed Ping

- Sends windows full of ICMP Echo or UDP Port Unreachable packets
- Shows packet throughput and loss under varying load (window sizes)

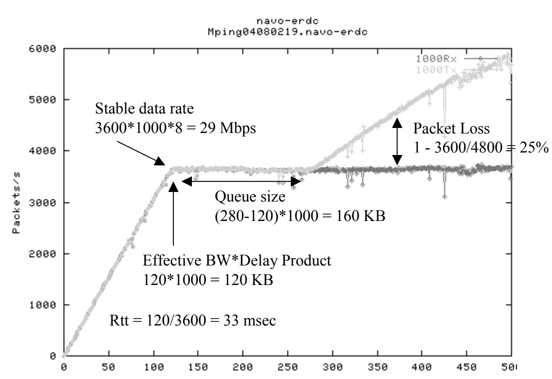
Example: window size = 5



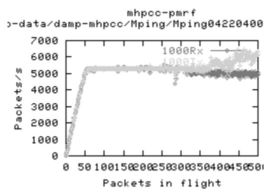
MPing on a “Normal” Path

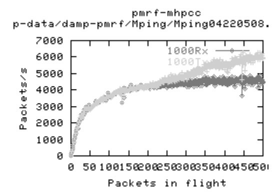


MPing on a “Normal” Path



Some MPing Results #1

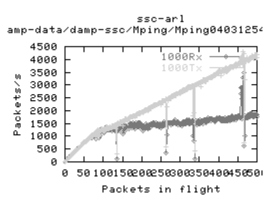


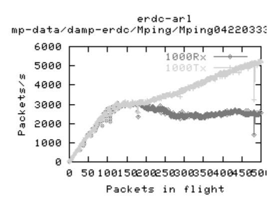


Fairly normal behavior
Discarded packets are costing some performance loss

RTT is increasing as load increases
Slow packet processing?

Some MPing Results #2

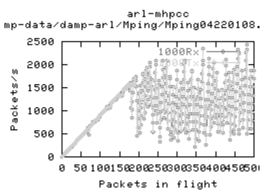


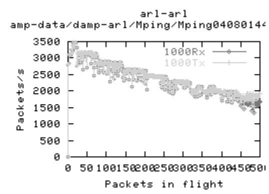


Very little stable queueing
Insufficient memory?
Spikes from some periodic event (cache cleaner?)

Discarding packets comes at some cost to performance
Error logging?

Some MPing Results #3

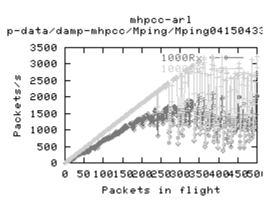


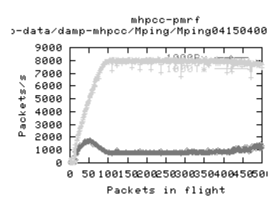


Oscillations with little loss
Rate shaping?

Decreasing performance with
increasing queue length
Typical of Unix boxes with
poor queue insertion

Some MPing Results #4





Fairly constant packet loss,
even under light load

Major packet loss, ~7/8 or 88%
Hump at 50 may be duplex problem

Both turned out to be an auto-negotiation duplex problem
Setting to static full-duplex fixed these!

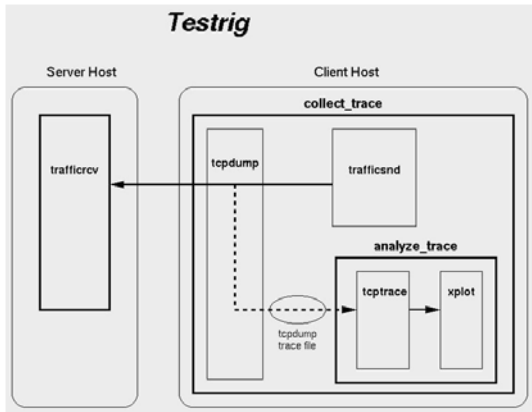
Advanced Debugging

TCP Traces and Testrig

TCP/IP Analysis Tools

- tcpdump
  - www.tcpdump.org
- ethereal - GUI tcpdump (protocol analyzer)
  - www.ethereal.com
- tcptrace – stats/graphs of tcpdump data
  - www.tcptrace.org
- testrig – tcpdump, tcptrace, xplot, etc.
  - www.ncne.nlanr.net/research/tcp/testrig/

“A Preconfigured TCP Test Rig”



tcptrace -l

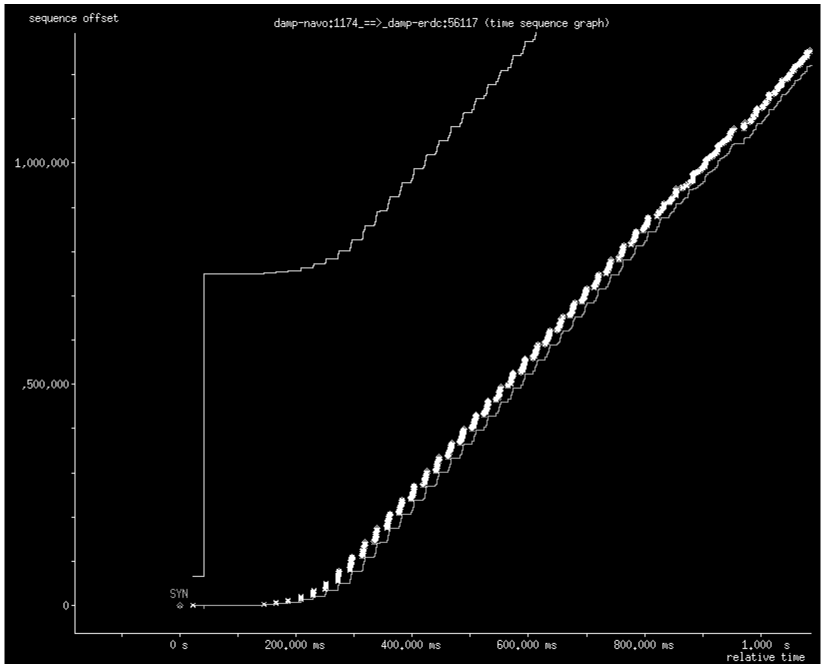
```
TCP connection 1:
        host a:        sd.wareonearth.com:1095
        host b:        amp2.sd.wareonearth.com:56117
        complete conn: yes
        first packet:  Sun Apr 23 23:35:29.645263 2000
        last packet:   Sun Apr 23 23:35:41.108465 2000
        elapsed time:  0:00:11.463202
        total packets: 107825
        filename:      trace.0.20000423233526
   a->b:                              b->a:
     total packets:         72032           total packets:         35793
     ack pkts sent:         72031           ack pkts sent:         35793
     pure acks sent:            2           pure acks sent:        35791
     unique bytes sent: 104282744           unique bytes sent:         0
     actual data pkts:      72029           actual data pkts:          0
     actual data bytes: 104282744           actual data bytes:         0
     rexmt data pkts:           0           rexmt data pkts:           0
     rexmt data bytes:          0           rexmt data bytes:          0
     outoforder pkts:           0           outoforder pkts:           0
     pushed data pkts:      72029           pushed data pkts:          0
     SYN/FIN pkts sent:       1/1           SYN/FIN pkts sent:       1/1
     req 1323 ws/ts:          Y/Y           req 1323 ws/ts:          Y/Y
     adv wind scale:            0           adv wind scale:            4
     req sack:                  Y           req sack:                  N
     sacks sent:                0           sacks sent:                0
     mss requested:          1460 bytes     mss requested:          1460 bytes
     max segm size:          1448 bytes     max segm size:             0 bytes
     min segm size:           448 bytes     min segm size:             0 bytes
     avg segm size:          1447 bytes     avg segm size:             0 bytes
     max win adv:           32120 bytes     max win adv:          750064 bytes
     min win adv:           32120 bytes     min win adv:           65535 bytes
     zero win adv:              0 times     zero win adv:              0 times
     avg win adv:           32120 bytes     avg win adv:           30076 bytes
     initial window:         2896 bytes     initial window:            0 bytes
     initial window:            2 pkts      initial window:            0 pkts
     ttl stream length: 104857600 bytes     ttl stream length:         0 bytes
     missed data:          574856 bytes     missed data:               0 bytes
     truncated data:    101833758 bytes     truncated data:            0 bytes
     truncated packets:     72029 pkts      truncated packets:         0 pkts
     data xmit time:       11.461 secs      data xmit time:        0.000 secs
     idletime max:          372.0 ms        idletime max:          246.8 ms
     throughput:          9097174 Bps       throughput:                0 Bps
```

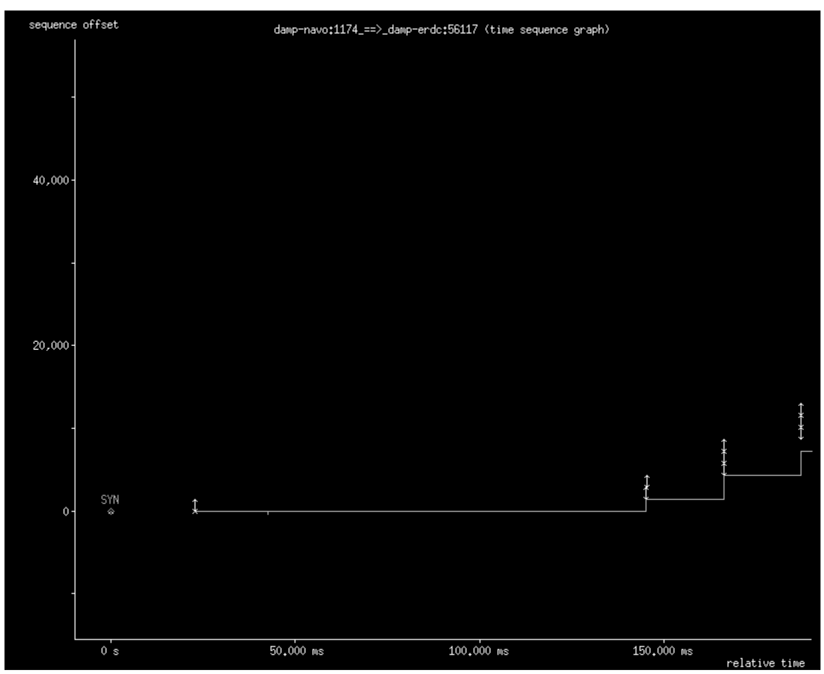
TCP Connection Establishment

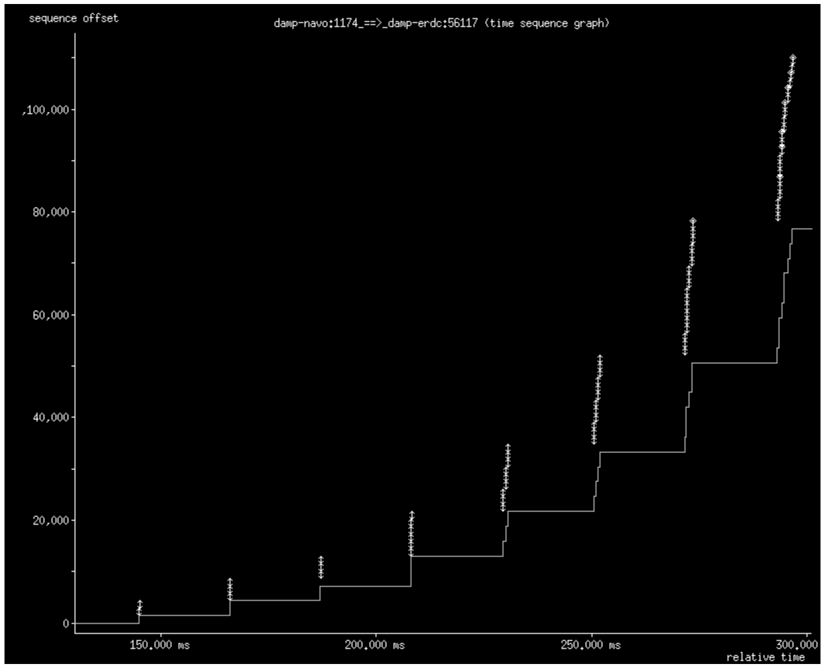
- Three-way handshake
  – SYN, SYN+ACK, ACK
- Use tcpdump, look for performance features
  – window sizes, window scale, timestamps, MSS, SackOK, Don't-Fragment (DF)

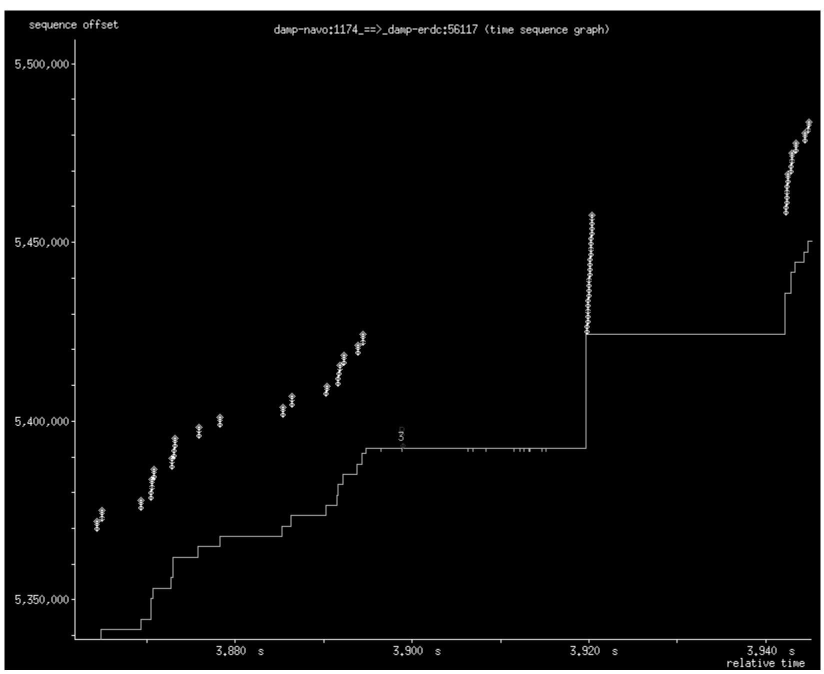
Tcpdump of TCP Handshake

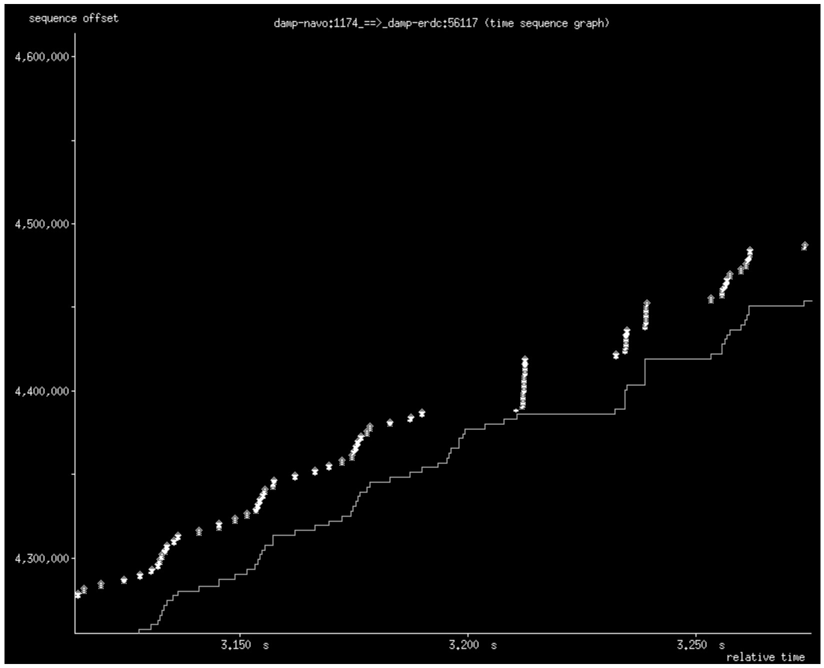
```
16:08:33.674226 wcisd.hpc.mil.40874 > damp-nrl.56117:
    S 488615735:488615735(0) win 5840
    <mss 1460,sackOK,timestamp 263520790 0,nop,wscale 0> (DF)

16:08:33.734045 damp-nrl.56117 > wcisd.hpc.mil.40874:
    S 490305274:490305274(0) ack 488615736 win 5792
    <mss 1460,sackOK,timestamp 364570771 263520790,nop,wscale 5> (DF)

16:08:33.734103 wcisd.hpc.mil.40874 > damp-nrl.56117:
    . ack 1 win 5840
    <nop,nop,timestamp 263520796 364570771> (DF)
```


sequence offset
damp-navo:1174_==>_damp-erdc:56117 (time sequence graph)
1,000,000
,500,000
0
SYN
0 s
200,000 ms
400,000 ms
600,000 ms
800,000 ms
1,000 s
relative time

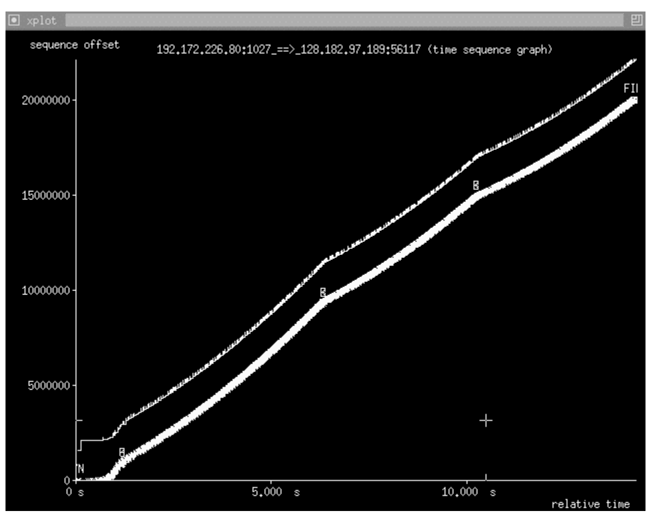

sequence offset
damp-navo:1174_==>_damp-erdc:56117 (time sequence graph)
40,000
20,000
0
SYN
0 s
50,000 ms
100,000 ms
150,000 ms
relative time


sequence offset
damp-navo:1174_==>_damp-erdc:56117 (time sequence graph)
,100,000
80,000
60,000
40,000
20,000
0
150,000 ms
200,000 ms
250,000 ms
300,000
relative time


sequence offset
damp-navo:1174_==>_damp-erdc:56117 (time sequence graph)
5,500,000
5,450,000
5,400,000
5,350,000
3.880 s
3.900 s
3.920 s
3.940 s
relative time

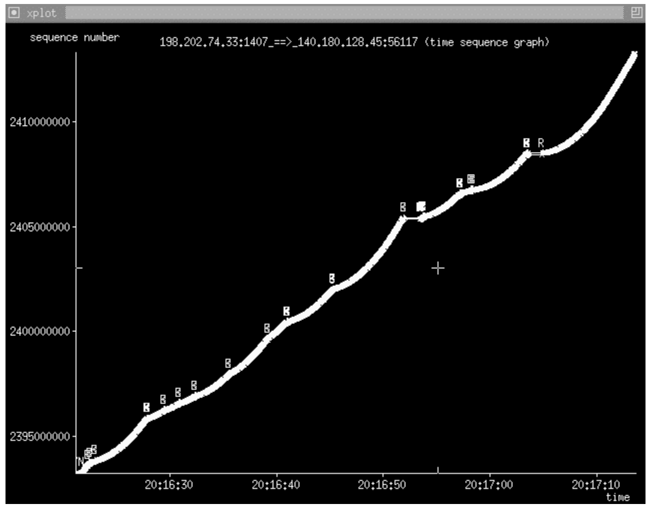


Normal TCP Scallops



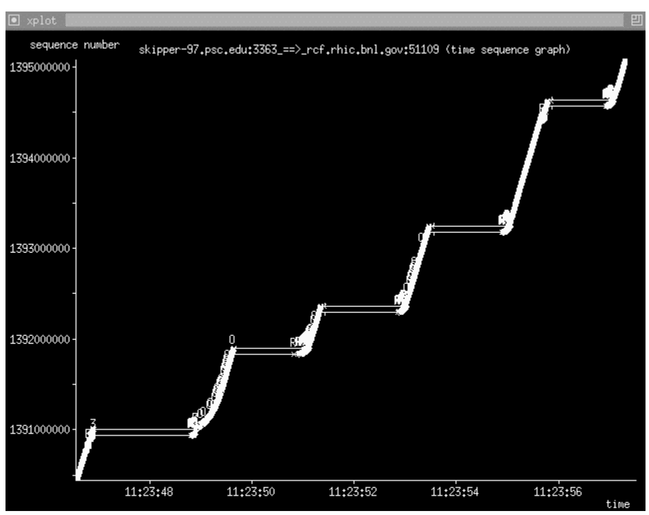
NLANR
NCNE

A Little More Loss



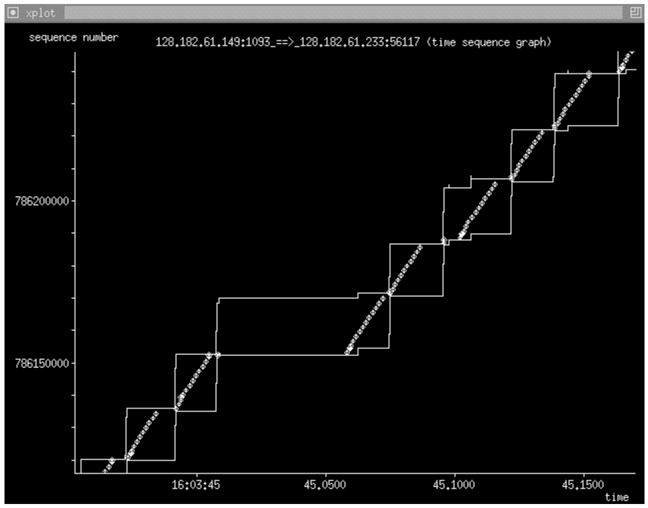
NLANR
NCNE

Excessive Timeouts



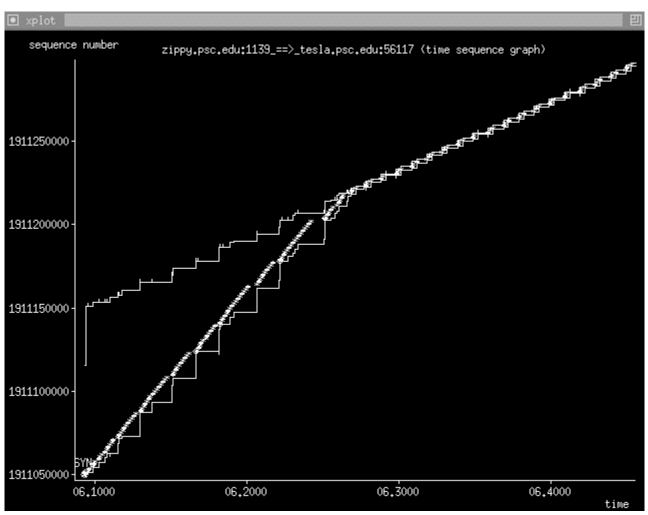
NLANR
NCNE

Bad Window Behavior



NLANR
NCNE

Receiving Host/App Too Slow



NLANR
NCNE

The Future of TCP/IP

- Different retransmit/recovery schemes
  - TCP Taho, Vegas, Peach, Westwood, …
- Pacing - removing burstiness by spreading the packets over a round trip time (BLUE)
- Rate-halving to recover ACK clocking more quickly
- Limited Transmit – open window on duplicate ACKs

The Future of TCP/IP cont.

- Receiver mods to prevent sender "cheating"
- Autotuning buffer space usage
- Kick-starting TCP after timeouts
- Explicit Congestion Notification (ECN)
- IPv6
- Multi Protocol Label Switching (MPLS)

Review

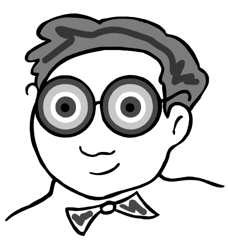
- Network capacity vs. speed
- Importance of window and buffer sizes
- How TCP throughput depends on delay, loss, packet size
- How to use ping, traceroute, treno, etc.
- Looking deeper for problems
- TCP/IP is still evolving

Recommended Resources

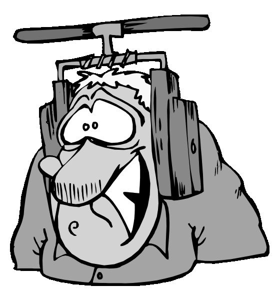
- Richard W. Stevens' books
  - TCP/IP Illustrated, ISBN 0-201-63346-9
  - http://www.kohala.com/start/
- Host performance tuning details
  - http://www.psc.edu/networking/perf_tune.html
- CAIDA Internet Measurement Tool Taxonomy
  - http://www.caida.org/tools/

Recommended Resources

- Iperf for TCP and UDP throughput testing
  - http://dast.nlanr.net/Projects/Iperf/
- Testrig for TCP traces
  - http://ncne.nlanr.net/research/tcp/testrig/



Thank You!



Phillip Dykstra
WareOnEarth Communications Inc.
2109 Mergho Impasse
San Diego, CA 92110
phil@sd.wareonearth.com
619-574-7796